

Serial No: 09/948,286

IN THE CLAIMS:

Please amend claims 1-10 as follows:

1 Claim 1. (currently amended) A method for operating an out-of-order
2 processor ~~in which a rename process is comprised of the pipeline an~~
3 instruction pipeline stream ~~is processed with~~, the method comprising the
4 steps of:

5 for detection of a dependency, determining for each current instruction
6 involved in a renaming process that a logic target address of one or more
7 instructions stored in a temporary buffer associated with a pipeline process
8 downstream of the current instruction is not the same as a logic source
9 address of said current instruction, said one or more instructions being
10 stored in a temporary buffer associated with a pipeline process downstream of
11 the current instruction;

12 generating a ~~no-dependency signal~~ no-dependency signal associated with
13 said current instruction; and

14 ~~forwarding said signal for exploiting said signal in order to control~~
15 ~~the processing of said current instruction in order to bypass a portion of~~
16 ~~the pipeline if the no-dependency signal is not active, assigning an entry in~~
17 ~~the temporary buffer to the logic source address of said current instruction;~~
18 and

19 if the no-dependency signal is active, issuing the instruction operand
20 data to an instruction execution unit without assigning the entry in the
21 temporary buffer to the logic source address of said current instruction.

1 Claim 2. (currently amended) The method according to claim 1 in which
2 the step of generating a no dependency signal comprises the steps of:

3 comparing a plurality of logic target register addresses and the logic
4 source register address of the current instruction, ~~in case of a match the~~
5 no-dependency signal is not active; and

6 generating a ~~dependency signal~~ dependency signal for the respective
7 source register.

1 Claim 3. (currently amended) The method according to claim 1 further
2 comprising the step of evaluating 'valid'-bits of ~~non-architected-state~~
3 speculative target registers stored in a storage associated with
4 speculatively calculated instruction result data to generate into the no-
5 dependency signal generation no-dependency signal.

Serial No: 09/948,286

1 Claim 4. (currently amended) The method according to claim 1 further
2 comprising the step of applying the method to a mapping table based renaming
3 scheme comprising the step steps of:

4 addressing a mapping-table-entry with a logical source register address
5 of said current instruction thus determining the mapped physical target
6 register address;

7 reading a ~~committed-status flag~~ committed-status flag in said entry;

8 comparing the logic target register address and the logic source
9 register address of the current instruction, ~~and in case of a match~~ the no-
10 dependency signal is not active; and

11 generating a ~~dependency~~ dependency signal for the respective
12 source register.

1 Claim 5. (currently amended) The method according to claim 2 further
2 comprising the step of applying the method to a mapping-table-based renaming
3 scheme comprising the step steps of:

4 addressing a mapping table entry with a logical source register address
5 of said current instruction thus determining the mapped physical target
6 register address;

7 reading a ~~committed-status flag~~ committed-status flag in said entry;

8 comparing the logic target register address and the logic source
9 register address of the current instruction, ~~and in case of a match~~ the no-
10 dependency signal is not active; and

11 generating a dependency-signal for the respective source register.

1 Claim 6. (currently amended) A processing system having means for
2 executing a readable machine language, said readable machine language
3 comprises:

4 a first computer readable code for, the detection of a dependency,
5 determining for each current instruction involved in a renaming process that
6 a logic target address of one or more instructions stored in a temporary
7 buffer associated with a pipeline process downstream of the current
8 instruction is not the same as a logic source address of said current
9 instruction,

10 a second computer readable code for generating a ~~no-dependency~~ no-dependency signal
11 no-dependency signal associated with said current instruction, ~~and~~

12 a third computer readable code for ~~forwarding~~ forwarding ~~said signal for~~
13 ~~exploiting~~ exploiting ~~said signal in order to control the preecessing of said current~~
14 ~~instruction in order to bypass a portion of the pipeline~~ assigning an entry

Serial No: 09/948,286

15 in the temporary buffer to the logic source address of said current
16 instruction if the no-dependency signal is not active; and
17 a fourth computer readable code for issuing the instruction operand
18 data to an instruction execution unit without assigning the entry in the
19 temporary buffer to the logic source address of said current instruction if
20 the no-dependency signal is active.

1 Claim 7. (currently amended) The processing system according to claim
2 6 in which in case of a content-addressable memory (CAM)-based renaming
3 scheme the first computer readable code for determining the dependency of a
4 current instruction comprises a compare logic in which all instructions to be
5 checked for dependency are involved and ~~a post-connected an OR gate coupled~~
6 with the compare logic.

1 Claim 8. (currently amended) The processing system according to claim
2 7 further comprising a plurality of AND gates the input of which comprises a
3 ~~the target register 'valid-bits'~~ 'valid bits' signal and a respective compare
4 logic output signal.

1 Claim 9. (currently amended) The processing system according to claim
2 6 in which the case of a mapping-table-based renaming scheme each mapping
3 table entry comprises an additional instruction-committed flag, and the first
4 computer readable code for determining the dependency of a current
5 instruction comprises a logic for ANDing ~~a selected a target register 'valid~~
6 bits' signal in which all instructions to be checked for dependency are
7 involved and ~~a post-connected an OR gate coupled with the logic.~~

1 Claim 10. (currently amended) A computer system having an out-of-order
2 processing system, said computer system executes a readable machine language,
3 said readable machine language comprises:

4 a first computer readable code for, the detection of a dependency,
5 determining for each current instruction involved in a renaming process that
6 a logic target address of one or more instructions stored in a temporary
7 buffer associated with a pipeline process downstream of the current
8 instruction is not the same as a logic source address of said current
9 instruction,

10 a second computer readable code for generating a ~~no-dependency-signal~~
11 no-dependency signal associated with said current instruction, and
12 a third computer readable code for ~~forwarding said signal for~~
13 ~~exploiting said signal in order to control the processing of said current~~
14 ~~instruction in order to bypass a portion of the pipeline assigning an entry~~

Serial No: 09/948,286

15 in the temporary buffer to the logic source address of said current
16 instruction if the no-dependency signal is not active; and
17 a fourth computer readable code for issuing the instruction operand
18 data to an instruction execution unit without assigning the entry in the
19 temporary buffer to the logic source address of said current instruction if
20 the no-dependency signal is active.